AMENDMENTS TO THE SPECIFICATION

Applicant respectfully requests that the following, amended paragraphs replace the corresponding paragraphs in the specification. The amendments correct typos and do not add new matter.

[0006] Two approaches are typical in the art for testing a code segment. In a first approach, a programmer manually creates a test program, compiles the program, and then executes the program to investigate the behavior. Although logically effective for testing the code segment, this approach is less than optimal for time and space efficiency for several reasons. One, this approach is error prone because a new source file must be created and imports are manually resolved. Additionally, the steps within this approach add duration to code development. Finally, this approach clutters the working environment with additional test programs to manage, use to incorporate knowledge into the original source program, and delete without losing modifications. Even if a programmer does not use additional source files to test, the programmer exerts more effort to keep the initial work environment clean and clear by deleting all test statements such as print statements of values of variables near any variable[[s]] reassignments in the code.

[0010] In addition, these debugging features require the source file to have an entry point. An entry point is a point in code that indicates where to begin compiling or executing the source file. For example, code written in C includes an "{" and[["{"]] "}" to delineate the main program. Accordingly, execution begins at the entry point, the "{", and code statements direct the processing flow subsequently until flow ends at the "}". Because WSAD's inspect feature requires an entry point, the inspect feature does not aid a programmer who would like to inspect how a function within a standard library operates. In this example, to use the inspect feature to test a library function, the programmer would write a separate testing program to meet the requirements of being compilable and executable and containing an entry point; then, the programmer would use the separate testing program to call the library function. An additional drawback of the WSAD inspect feature is that this approach requires running the entire program

if the source program is not self-executable. Ultimately, this approach focuses on the testing phase of the code life cycle rather than the development or coding phase.

[0019] In many embodiments, hardware and/or software may implement circuit structure logic to test a segment of programming code. The hardware and/or software may, for instance, create a temporary source file. Through this method, a temporary source file may include and test a code segment. The temporary source file may also optionally insert an entry point and copy[[ing]] external code referenced by the code segment into the temporary source file.

[0038] FIGs 5A-C depict an example of a code segment 508 of Java™ code together with the source file 502 that contains the code segment to be tested, an embodiment of an associated temporary source file 504, and generated output 506. In this embodiment, source file 502 begins with a comment line 510 indicating its title, "// javaexample.java". Next, source file 502 includes lines 512 and 514, "import java.io.*;" and "import java.util.*;". Lines 512 and 514 reference the standard libraries, java.io and java.util to provide access to functions defined in those libraries. Then, line 516 defines a public class javaexample; and line 518 defines a public main function that may input string arguments and returns a null value. Following in the source file 502, in line 520, a "{" represents the entry point of the main function. Next, in line 522, source file 502 defines offset1str and offset2str as string type variables. In line 524, source file 502 defines offset1 and offset2 as double type variables. In line 526, source file 502 defines database as an array variable including 10 double elements within it. Then, in line 528, source file 502 defines x, y, and n as double variables. In line 530, with "rect (x, y, 200, 200);", source file 502 would attempt to draw a box; however, x and y are not yet defined. In line 532, with "fill (x, y, 50, 240);", source file 502 would attempt to fill in color inside the box; however, x and y are still not yet defined. In line 534, "for n=1 to 10" begins a for loop to repeat the contents between the "{" and the "}" in lines 536 and 538 for 10 times. Within the for loop, line 538 which reads, "{database (n)=n", assigns elements of the database array as sequential numbers 1 through 10. The last lines 542-552 appear shadowed inside a shaded area 508; the programmer selected these lines for Evaluate Segment Function 116 to test. In another embodiment, the programmer may shade lines of code that are not sequential; for example, a code segment may

include lines of code dispersed through a source file in nonadjacent lines by pressing the Control key while highlighting lines with the Shift key using a GUI.